REMARKS

This Amendment is submitted in response to the Office Action dated November 27, 2007, having a shortened statutory period set to expire February 27, 2008. Proposed amendments to the Claims include amending Claims 1, 3-6, 10, 12, 14-17, 19-20 and 22, and cancelling Claim 21. Upon entry of the proposed amendments, Claims 1, 3-6, 10, 12, 14-17, 19-20 and 22 will now be pending.

Applicants' undersigned representative appreciates the time and courtesy extended by the Examiner during a November 21, 2007 teleconference. No formal agreement was reached regarding the patentability of the pending claims. During that teleconference, the Examiner suggested that references to "Java" in the specification include appropriate trademark notices. Attached as appendices to the present amendment are a "clean" and "marked up" version of a replacement specification making the suggested changes.

REJECTIONS UNDER 35 U.S.C. § 112

In paragraph 3 of the present Office Action, the Examiner has rejected Claims 1, 3-6, 10, 12, 14-16 and 21-22 for using the indefinite term "Java." The current amendment removes all references to "Java," and substitutes in either no limiter or a reference to a condensed interpreted language, as supported in the original specification on page 5, lines 22-27, and on page 18, line 4 to page 20, line 4. Applicants therefore respectfully request that these rejections be withdrawn.

REJECTIONS UNDER 35 U.S.C. § 101

In paragraph 4 of the present Office Action, the Examiner has rejected Claims 10, 12 14-17 and 19 under 35 U.S.C. § 101, for claiming a possibly non-tangible computer medium. The present amendment uses the Examiner-suggested language of a "tangible computer-readable medium storing computer program code." Applicants therefore request that these rejections be withdrawn.

REJECTIONS UNDER 35 U.S.C. § 103

In paragraph 5 of the present Office Action, Claim 20 is rejected under 35 USC 103(a) as being unpatentable over *Schwabe* (U.S. Patent No. 6,883,163 – "*Schwabe*") in view of *Stammers et al.* (U.S. Patent No. 7,069,554 – "*Stammers*") and in view of *Ji* (U.S. Patent No. 6,272,641 – "*Ji*") and in view of *Levy et al.* (U.S. Patent No. 6,092,147 – "*Levy*").

The Examiner cites Figures 4-6 and column 6, lines 60-67, column 7, lines 1-3 and column 7, lines 10-15 of *Schwabe* for teaching the claimed feature of "converting an original file into a reduced file, wherein the original file contains a <u>class description section</u> and an <u>instruction section</u>, and wherein the reduced file contains a <u>code description section</u> that is based on the class description section, and wherein the reduced file contains a <u>code section</u> that is based on the instruction section, wherein the original file contains classes that are capable of being compiled, and <u>wherein the only executable instructions in the reduced file are applets</u>."

The cited figures show:

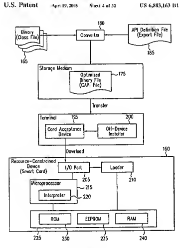

FIG. 4
(PRIOR ART)

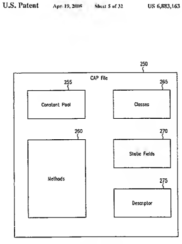

FIG. 5
(PRIOR ART)

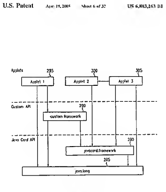

FIG. 6
(PRIOR ART)

The cited passages state:

> Referring to FIG. 4, development of an applet for a resource-constrained device, such as a smart card 160, begins in a manner similar to development of a Java.TM. program. In other words, a developer writes one or more Java.TM. classes and compiles the source code with a Java.TM. compiler to produce one or more class files 165. The applet can be run, tested and debugged, for example, on a workstation using simulation tools to emulate the environment on the card 160. When the applet is ready to be downloaded to the card 160, the class files 165 are converted to a converted applet (CAP) file 175 by a converter 180. The converter 180 can be a Java.TM. application being executed by a desktop computer. The converter 180 can accept as its input one or more export files 185 in addition to the class files 165 to be converted. An export file 185 contains naming or linking information for the contents of other packages that are imported by the classes being converted.

> Referring to FIG. 5, the CAP format is parallel to the class file information. Each CAP 250 contains all of the classes and interfaces defined in one Java.TM. package. A CAP file 250 has a compact and optimized format, so that a Java.TM. package can be efficiently stored and executed on resource-constrained devices. Among other things, the CAP file 250 includes a constant pool component (or "constant pool") 255 that is packaged separately from a methods component 260.

The cited figures and passage merely describe development of applets for smart cards, but do not teach or suggest "converting an original file into a reduced file, wherein the original file contains a class description section and an instruction section, and wherein the reduced file contains a code description section that is based on the class description section, and wherein the reduced file contains a code section that is based on the instruction section, wherein the original file contains classes that are capable of being compiled, and wherein the only executable instructions in the reduced file are applets."

The Examiner cites column 7, lines 21-26 of *Stammers* for teaching the claimed feature of "converting the reduced file into a converted file, wherein the reduced file and the converted file are semantically identical." The cited passage from *Stammers* states:

> The classloader object for the control JAR file thus created can then be used to read classes from the JAR file object created at step S6 and to create objects therefor in the Java virtual machine 32 by converting the class byte code into executable code in Java virtual machine working memory 42.

The cited passage merely describes the creation of executable code. There is no suggestion of a "reduced file" and a "converted file," particularly two files that are semantically identical.

Furthermore, there is no teaching or suggestion of the features of "wherein the converted file is created by a preconversion step and a mapping step, wherein the preconversion step converts identifier names in the reduced file into symbolic names to generate a preconverted file that comprises a preconverted code section and a preconverted code description section, and wherein the mapping step replaces the symbolic names with original names used by the original file to create the converted file," as supported in the original specification on page 18, line 28 to page 19, line 24.

The Examiner cites Figure 2 and column 8, lines 1-4 of *Ji* for teaching the feature of "creating a cryptographic signature for the converted file." Figure 2 shows:

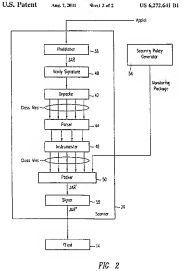

FIG. 2

Column 8, lines 1-4 of *Ji* states:

> The digital signer component 58 digitally signs the applet (now JAR"), with a digital signature unique to the particular scanner 26, for authentication in the local domain. The applet JAR" is then transferred to the client machine 14 for execution.

Thus, *Ji* teaches that a Java Archive (JAR) file can be digitally signed. However, a JAR file contains only original files (see *Ji* column 7, lines 8-17), NOT converted files, particularly converted files that have been created in the two-step manner claimed.

With regards to the feature of "storing the cryptographic signature and the reduced file in a chipcard, wherein the cryptographic signature verifies that the reduced file was converted by a trusted entity," the Examiner states that *Levy* teaches this feature in Figures 4 and 5, and at column 6, lines 11-27. The cited passage states:

> The front end bytecode verifier 68 may verify that one or more bytecodes entering the converter from source outside of the converter, such as compilers or other forms of software application generators, conform to a predetermined set of criteria. The criteria may be similar to the verification steps described above with reference to FIG. 2. Any bytecodes which do not conform to the criteria may be rejected. The resulting verified bytecodes may be transferred to the bytecode authenticator 70. The bytecode authenticator may receive bytecodes exclusively from the bytecode front end verifier and may compute and generate a proof of authenticity, as is well known, on the one or more verified bytecodes using on any suitable cryptographic computation. A suitable cryptographic computation may include, for example, a hash value, a message authentication code using a block-cipher algorithm, or a digital signature using an asymmetric cryptographic algorithm.

Figures 4 and 5 of *Levy* show:

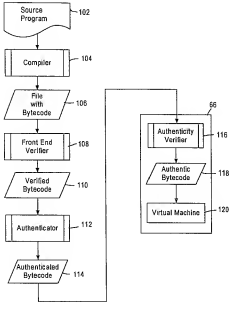

FIG. 4

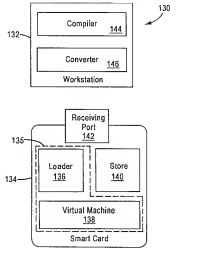

Fig. 5

There is no suggestion in any of the referenced and cited passages/figures for "storing the cryptographic signature and the reduced file in a chipcard." Rather, the cited passages merely show a virtual machine in a smart card (Figure 5), which can use authentic bytecode (Figure 4) as confirmed by cryptographic signatures (column 6, lines 11-27).

Applicants therefore respectfully request that the rejection of Claim 20 be withdrawn.

CONCLUSION

As a combination of the cited art does not teach or suggest all limitations found in the presently pending claims, Applicants now respectfully request a Notice of Allowance for all pending claims. If the Examiner believes that a teleconference would be useful in promoting any or all of the present claims to allowance, such a telephone call to the Applicant's undersigned representative, at 512.617.5533, would be greatly appreciated.

No extension of time for this response is believed to be necessary. However, in the event an extension of time is required, that extension of time is hereby requested. Please charge any fee associated with an extension of time as well as any other fee necessary to further the prosecution of this application to **IBM CORPORATION DEPOSIT ACCOUNT No. 09-0461**.

Respectfully submitted,

James E. Boice
Registration No. 44,545
DILLON & YUDELL LLP
8911 North Capital of Texas Hwy.
Suite 2110
Austin, Texas 78759
512.617.5533

ATTORNEY FOR APPLICANT(S)